# Технический протокол

## к Договору на оказание услуг

## Описание интерфейса стандартного взаимодействия с Процессинговой системой

1. Интерфейс Партнера должен обеспечивать:
   - прием запросов по протоколу HTTPS с IP-адресов подсети: 194.187.247.152
   - обработку параметров, передаваемых системой методом GET.
   - формирование ответа Процессинговой системе (далее – система) в формате XML в кодировке UTF-8 (если ответ содержит символы национальных алфавитов).
2. Обмен информацией осуществляется в режиме «запрос-ответ». Система разрывает соединение по таймауту если скорость ответа превышает 15 секунд.
3. Если предполагаемое количество Платежей, подключаемых Партнером, ожидается интенсивное (более 10 Платежей в минуту), необходимо, чтобы интерфейс поддерживал многопоточную коммуникацию до 10-15 одновременных соединений.
4. В случае отсутствия интерфейса у Партнера взаимодействие с системой осуществляется согласно формату электронных файлов с данными. Формат электронных файлов согласовывается через Каналы связи.
5. Платежное приложение Партнера payment_app.cgi, располагается по адресу kaspi.partner.kz, сервер поддерживает HTTPS соединения на порт 443.
6. Информация о Платежах передается в следующем формате:

| № п/п | Переменная | Описание |
|---|---|---|
| 1 | **txn_id** | Номер Платежа в системе, который передается Партнеру, целое число длиной до 18 знаков |
| 2 | **prv_txn** | Уникальный номер Платежа пополнения баланса Плательщика (в базе Партнера), целое число длиной до 20 знаков. Этот элемент должен возвращаться Партнером после запроса на пополнение баланса (запроса «pay»). При ответе на запрос на проверку состояния счета Плательщика (запрос «check») его возвращать не нужно – не обрабатывается |
| 3 | **sum** | Сумма Платежа, передаваемая Партнеру. Дробное число с точностью до сотых, в качестве разделителя используется «.» (точка). Если сумма представляет целое число, то оно все равно дополняется точкой и нулями, например – «200.00» |
| 4 | **txn_date** | Дата в формате ГГГГММДДЧЧММСС. Эту дату необходимо использовать для проведения бухгалтерских взаиморасчетов |
| 5 | **account** | Идентификатор Плательщика (номер лицевого счета, телефона, логин и т.д.). Строка, содержащая буквы, цифры и спецсимволы, длиной до 200 символов |
| 6 | **command** | Строка, принимающая значения «check» и «pay» |
| 7 | **command=pay** | Запрос на пополнение баланса Плательщика |
| 8 | **command=check** | Запрос на проверку состояния счета Плательщика. В запросе check значение в переменной sum является номинальным , передается с терминала по умолчанию и его обрабатывать не нужно |
| 9 | **response** | Тело ответа |
| 10 | **result** | Код результата завершения запроса |
| 11 | **comment** | Необязательный элемент – комментарий завершения операции |
| 12 | **data1,data2,…,dataN** | Дополнительные поля, передаваемые Партнером - строки, содержащие буквы, цифры и спецсимволы |
| 13 | **field1, field2… fieldN** | Поля, где содержится информация, которую необходимо передать системе. Эта информация может быть показана Партнеру при совершении Платежа. |

| | | |
|---|---|---|
| 14 | **fields** | Дополнительная информация о платеже |

7. Требования к порядку проведений Платежей:
   7.1. Сверка взаиморасчетов и решение спорных вопросов производятся на основании номера Платежа, в переменной **txn_id**.
   7.2. Сумма Платежа принимается от Плательщика и передается Партнеру в тенге в переменной **sum**.
   7.3. В запросе на добавление Платежа система передает дату Платежа в переменной **txn_date**. Учет платежей в системе Оператора и расчеты с Партнером ведутся по дате получения запроса от Плательщика. Чтобы избежать несоответствий при проведении сверок система передает Партнеру дату, по которой нужно учитывать Платеж.
   7.4. Партнер идентифицирует Плательщика по уникальному идентификатору. Перед отправкой Партнеру, идентификатор проходит проверку корректности в соответствии с регулярным выражением, которое должен предоставить Партнер. Идентификатор Плательщика передается в переменной **account**.

Возможна передача дополнительных полей в строке запроса data1,data2,…,dataN.

   7.5. Передача информации о Платеже Партнеру производится системой в 2 этапа:
      1) проверка состояния счета Плательщика При проверке состояния счета (запрос «check»), Партнер должен проверить наличие в своей базе Плательщика с указанным идентификатором и выполнить внутренние проверки идентификатора, суммы Платежа в соответствии с принятой логикой пополнения лицевых счетов через платежные системы.
      Результат запроса «result=0» означает, что лицевой счет Плательщика может быть пополнен.
      При необходимости, дополнительную информацию о Платеже можно передать в теге fields.
      2) проведение Платежей. При проведении Платежа (запрос «pay») Партнер должен произвести пополнение баланса Плательщика.
      Результат запроса «**result=0»** означает успешное завершение Платежа по пополнению баланса. Система полностью завершает обработку данного Платежа.
   7.6. В случае если любой из запросов к Партнеру завершается ошибкой, то Партнер возвращает код ошибки.

## Возможные коды состояния/ошибки запросов

| | |
|---|---|
| 0 | Плательщик/счет/заказ найден и доступен для пополнения/оплаты |
| 1 | "Плательщик/счет не найден" или "заказ не найден", если запрос «check» был на проверку состояния заказа |
| 2 | Заказ отменен |
| 3 | Заказ уже оплачен |
| 4 | Платеж в обработке |
| 5 | Другая ошибка Партнера |

7.7. Если система повторно присылает запрос с уже существующим в базе Партнера номером Платежа, то Партнер должен вернуть результат обработки предыдущего запроса.
Партнер возвращает ответ на запросы системе в формате XML со следующей структурой:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <txn_id></txn_id>
  <prv_txn></prv_txn>
  <sum></sum>
  <result></result>
  <comment></comment>
</response>
```

**Ответ Партнера формируется по следующей структуре:**

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <txn_id>1234567</txn_id>
  <result>0</result>
  <fields>
    <field1 name="name1"> value1</field1>
    <field2 name="name2"> value2</field2>
    <fieldN name="nameN"> valueN</fieldN>
  </fields>
  <comment></comment>
</response>
```

8. В информационном обмене между Оператором и Партнером участвуют:
   - со стороны Оператора – _____
   - со стороны Партнера - уполномоченные подразделения Партнера
9. Оператор информирует Партнера о проведенных Платежах:
   1) В Личном кабинете в режиме реального времени (при наличии технической возможности);
   2) Путем направления уведомления на электронный адрес, указанный в Заявлении (при наличии технической возможности);
   3) Путем предоставления реестра Зачислений по Каналам связи;
   4) Путем интеграции систем Партнера и Оператора (при наличии технической возможности).

**Описание сервиса Отмена Платежей**

Refund API

Описание протокола взаимодействия внешней информационной системы с платежным модулем kaspiPay в части отмены Платежей.

API взаимодействует по протоколу http, данные запроса/ответа передаются методом httpPost в Json сериализованном виде.

## 1. Операция подачи заявки на отмену

Адрес вызова: returnApi/Refund/RefundRequest

**Запрос:**

```
{
        "PaymentId":202001211095,
        "ReturnAmount": 10000,
        "RefundIdentificatior": "XGD-UUH-876",
        "Reason": "Частичный возврат средств клиенту"
}
```

**Описание запроса:**

PaymentId - Идентификатор Платежа в системе kaspiPay, который передается при совершении оплаты во внешнюю систему

RetyrnAmount - Сумма, подлежащая возврату по данному Платежу, в тиынах

RefundIdentificatior - Уникальный идентификатор отмены (генерируется Партнером, т.к. в рамках одного Платежа может быть более 1 возврата)

Reason - Причина возврата, заполняется Партнером, может использоваться при уведомлении Плательщика банка о возврате денег на счет

**Модель запроса**

```
public class RefundRequest
    {
        [Required(ErrorMessage = "PaymentId required")]
        public long PaymentId { get; set; }
        [Required(ErrorMessage = "ReturnAmount required")]
        public long ReturnAmount { get; set; }
        [Required(ErrorMessage = "RefundIdentificatior required")]
```

```
        [StringLength(maximumLength: 100, MinimumLength = 1, ErrorMessage = "RefundIdentificatior
lenght (1,100)")]
        public string RefundIdentificatior { get; set; }
        [Required(ErrorMessage = "Reason required")]
        [StringLength(maximumLength:500, MinimumLength = 1,ErrorMessage = "Reason lenght
(1,500)")]
        public string Reason { get; set; }
    }
```

**Ответ:**

В случае успешного создания заявки на отмену в ответ вернется структура

```
{
  "statusCode": 0,
  "requestIdentificatior": "414e2f7a-356d-4d0b-91f0-a3cf34717f2e"
}
```

В случае ошибки вернется структура:

```
{
  "statusCode": 1,
  "error": {
    "code": 4,
    "errorMessage": "Заявка уже существует. Проверьте статус"
  },
  "requestIdentificatior": "414e2f7a-356d-4d0b-91f0-a3cf34717f2e"
}
```

**Модель ответа**

```
 public class RefundResponse
    {
        public RefundStatus StatusCode { get; set; }
        public OperationError Error { get; set; }
        public string RequestIdentificatior { get; set; }
    }
 public class OperationError
   {
      public int Code { get; set; }
      public string ErrorMessage { get; set; }
   }

 public enum RefundStatus
    {
        ACCEPT = 0, //Заявка принята в работу, запросите статус позднее(нетерминальный)
        DECLINE = 1, //Заявка не принята, причина в структуре OperationError(смотреть ошибку)
        SUCCESS = 2, //Возврат средств клиенту успешно произведен, заявка
выполнена(терминальный)
        FAILED = 3, //Заявка упала в ошибку, причина в структуре OperationError(терминальный)
        INPROGRESS = 4, //Заявка обрабатывается, результат будет известен
позже(нетерминальный)
        FATAL= 5, //Фатальная ошибка(неизвестный)
            MANUAL = 6, //Заявка обрабатывается в ручном режиме, статус будет известен позже
(нетерминальный)
        INPROGRESS_ERROR =7 //Внутренний статус, используется для перехода из
автоматической вручную обработку (нетеминальный)
    }
```

Обработка заявки может проходить в ONLINE и OFFLINE автоматических и в ручном режиме
Режим выбирается системой платежей kaspiPay в момент создания заявки на отмену, и зависит от множества факторов.
В случае ONLINE автоматического режима запрашивающей системе сразу выдается конечный статус (2 или 3)
В случае OFFLINE автоматического режима запрашивающей системе выдается результат (0) и requestIdentificatior - уникальный идентификатор отмены, по которому позже
можно будет запросить статус
В случае ручного режима выдается такой-же ответ как в случае OFFLINE автоматического режима.

## 2. Операция запроса статуса заявки на отмену

**Адрес вызова:** returnApi/Refund/RefundStatus
**Запрос:**
```
{
        "RequestIdentificatior":"414e2f7a-356d-4d0b-91f0-a3cf34717f2e"
}
```
**Модель запроса**
```
 public class StatusRequest
      {
         [Required(ErrorMessage = "RequestIdentificatior required")]
         public string RequestIdentificatior { get; set; }
      }
```
**Успешный ответ**
```
{
  "statusCode": 0
}
```

**Неуспешный ответ:**
```
{
  "statusCode": 5,
  "error": {
    "code": 6,
    "errorMessage": "Заявка на отмену не найдена"
  }
}
```
**Модель ответа:** используется та же модель, что и на метод создания заявки на отмену

Варианты "statusCode" описаны в описании предыдущего метода

**Варианты ошибок:**
Code = -1, ErrorMessage = "Общая ошибка обработки"
Code = -2, ErrorMessage = "Фатальная ошибка обработки"
Code = 1, ErrorMessage = "Оригинальная транзакция не найдена"
Code = 2, ErrorMessage = "Сумма отмены превышает сумму оригинальной операции"
Code = 3, ErrorMessage = "Отмена невозможна. Статус операции не является отменяемым"
Code = 4, ErrorMessage = "Заявка уже существует. Проверьте статус"
Code = 5, ErrorMessage = "Создать заявку невозможно, по платежу есть активная или ошибочная заявка"
Code = 6, ErrorMessage = "Заявка на отмену не найдена"

**Авторизация**
**Адрес вызова:** returnApi/Auth/GetToken
**Запрос:**
```
{
        "Login":"userLogin",
        "Password":"123456"
}
```
**Ответ:**
**Успешная авторизация:**
HTTP STATUS CODE 200
```
{
  "token": "979df876-c3a3-4052-bea9-47d3f38eaa39",
  "validTo": "2020-03-19T18:05:07.902529+06:00"
}
```
**Неуспешная авторизация**
HTTP STATUS CODE 401
```
{
  "message": "Авторизация неуспешна"
}
```

**Проверка токена**
**Адрес вызова:** http://localhost:50258/returnApi/Auth/CheckToken

**Ответ токен валидный:**
HTTP STATUS CODE 200

{
"token": "979df876-c3a3-4052-bea9-47d3f38eaa39",
"validTo": "2020-03-19T18:05:07.902529+06:00"
}
Ответ токен невалидный или просроченный
HTTP STATUS CODE 401
{
"message": "Авторизация неуспешна"
}
Алгоритм работы авторизации:
В случае успешного вызова метода getToken, возвращается значение token и его lifeTime
Токен является непродляемым идентификатором сессии. В случае истечения token, вызывающая система должна получить новый токен
В один момент времени вызывающая система может иметь только 1 активный токен, предыдущий токен становится недействительным
с момента выдачи вызывающей системе нового token.
Полученный токен вставляется в header http request с именем "token"
В случае получения http ошибок 401 и 403 вызывающей системе необходимо получить новый токен, и продолжить взаимодействие

**Контактные данные Оператора**
Электронный адрес Pay@kaspi.kz
Номер телефона отдела сопровождения партнеров: 8 (727) 3306231